



ORCA™ Series MODBUS Kinematics Triggering and Configuring

User Guide 221105

Version 1.1

Contents

REVISION HISTORY	3
Kinematic Controller	3
Software Triggering.....	3
Hardware Triggering.....	3
Configuring a Motion.....	4
IrisControls4™ Pages	5
Position	5
Kinematic.....	5
Configuring the Kinematic Controller Over Modbus.....	6
Example of QModMaster Setup	6
Example Set up Using Hercules	7
CRC Bytes.....	7
Relevant Registers	7
Reading a Register.....	8
Setting the Number of Kinematic Motions.....	8
Setting the Target Positions.....	8
Setting the Motion Duration.....	9
Setting the Chain Delay.....	9
Enabling Chaining and Setting the Motion Type	9
Configuring an Entire Motion in a Single Frame.....	9
Saving Kinematic Configurations.....	10
Enabling and Disabling Kinematic Mode.....	10
Triggering a Kinematic Motion	10
Clear Errors.....	10
Enabling Hardware Triggering	10
Troubleshooting	11
Messages not being received:.....	11
Motor not moving:	11
Motor is overshooting position targets:.....	11

REVISION HISTORY

Version	Date	Author	Reason
1.0	November, 2022	rm, sj	Initial Release
1.1	March, 2023	rm, ab	Additional Examples, formatting, troubleshooting section. Indicating which features are 6.1.6. Formatting

Kinematic Controller

Orca Series motors are equipped with a kinematic controller that provides motion profiles which allow movement to a shaft position over a specified period while respecting the chosen kinematic constraints. Types of kinematic motions available on the Orca Series include:

- 0 - Minimum power (linear acceleration).
- 1 - Maximum smoothness (minimum jerk).
- 2 - Minimum power (linear acceleration assuming 0 initial velocity). **(6.1.6)**
- 3 - Maximum smoothness (minimum jerk assuming 0 initial velocity). **(6.1.6)**

Up to 32 motions can be saved to a single Orca Series motor. Motions are configured either from the kinematic GUI page, or through direct writes to the motor's memory map. Motions can be initiated either by MODBUS messages, or by hardware edge detection.

Kinematic controller options are configured using the KIN_CONFIG register. This register controls the number of configured motions, as well as the motion triggering mechanism.

Register	Bits 15-8	Bits 7-6	Bit 5	Bits 4-0
KIN_CONFIG	Reserved	TRIG_PERIOD	HW_TRIG	NUM_MOTIONS

The NUM_MOTIONS field counts from 0b00000, indicating 1 configured motion, up to 0b11111 indicating 32 configured motions.

Software Triggering

When the KIN_CONFIG[HW_TRIG] bit is cleared, motions will be initiated by writing the desired motion number to the KIN_SW_TRIG[MOTION_ID] field. This register will be set to 0x1000 initially and will return to 0x1000 after processing a software trigger request. Software triggers of motions with a number higher than the value configured in KIN_CONFIG[NUM_MOTIONS] will be ignored.

Register	Bits 15-6	Bits 4-0
KIN_SW_TRIG	Reserved	MOTION_ID

Hardware Triggering

When the KIN_CONFIG[HW_TRIG] bit is set, motions will be initiated by driving the RX₂₋ pin on the RJ45 connector (pin 2) at a voltage at least 200 mV higher than the RX₂₊ pin (pin 1). The voltage difference must be held constant for the time specified in KIN_CONFIG[TRIG_PERIOD] before being released. If the appropriate amount of time was elapsed between edges, a hardware event will be triggered.

TRIG_PERIOD	Hold Period
0b00	0 ms
0b01	10 ms
0b10	50 ms
0b11	100 ms

If the TRIG_PERIOD field is changed, the kinematic configuration must be saved, and the motor must be power cycled to take effect.

A single hardware trigger event will queue a single motion, and any further motions that are chained together, up to the number stored in KIN_CONFIG[NUM_MOTIONS]. Once the final motion has completed, the next motion queued will be motion ID 0.

WARNING! Enabling hardware triggering will disable MODBUS communications.

Configuring a Motion

Individual kinematic motions consist of five variables:

- Motion target position (μm).
- Motion period (ms).
- Motion type (minimum power or maximum smoothness).
- Chain delay (delay in ms before triggering next motion if chain is enabled).
- Chain trigger (start next consecutive motion after this one).

The kinematic GUI page provides an interface for programming each of the motions, or the motions can be written directly to the memory map. A single motion configuration takes up six consecutive registers in the memory map, totaling 192 registers allocated for the motions (KIN_MOTION_0 to KIN_MOTION_31). The layout of each set of these registers is shown in table 2.

Offset from KIN_MOTION_#	Description
0	Position Target (Low 16 bits)
1	Position Target (High 16 bits)
2	Settling Time (Low 16 bits)
3	Settling Time (High 16 bits)
4	Chain Delay
5	Type & Chain

The motion type and chain trigger options occupy the same register with the following structure.

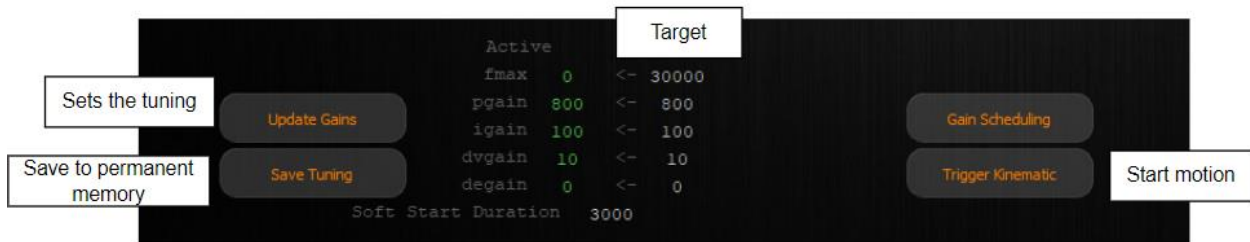
Register	Bits 15-3	Bit 2-1	Bit 0
KIN_MOTION_# + 5	Reserved	Type 00 = min. power 01 = min. jerk 02 = min. power ($V_{\text{init}} = 0$) (6.1.6) 03 = min. jerk ($V_{\text{init}} = 0$) (6.1.6)	Chain

The Type field is interpreted as a 2-bit binary number indicating motion type.

The Chain bit value is 1 when the chain feature is enabled and 0 when it is disabled.

IrisControls4™ Pages

Position



Active

fmax	0	<-	30000
pgain	800	<-	800
igain	100	<-	100
dvgain	10	<-	10
degain	0	<-	0

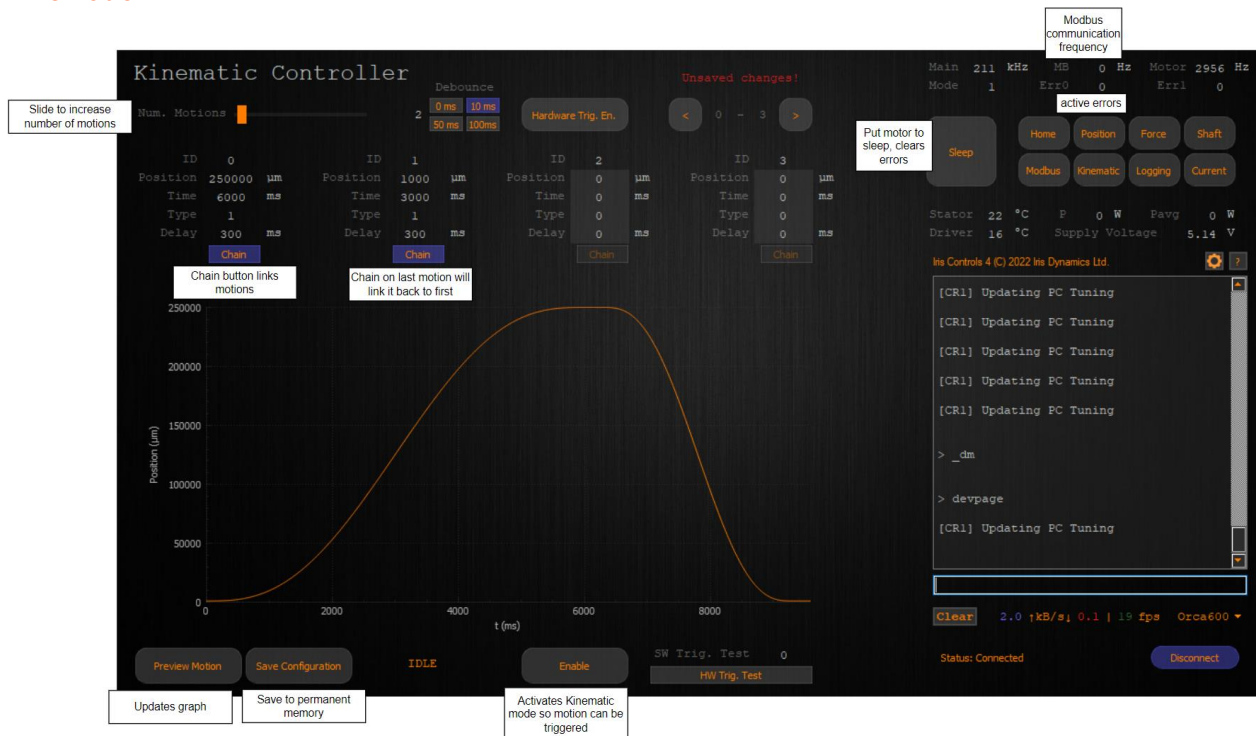
Soft Start Duration 3000

Target: 30000

Buttons: Update Gains, Gain Scheduling, Save Tuning, Trigger Kinematic

Annotations: Sets the tuning, Save to permanent memory, Start motion

Kinematic



Kinematic Controller

Unsaved changes!

Num. Motions: 2

ID	Position	Time	Type	Delay	Chain
0	250000 µm	6000 ms	1	300 ms	0
1	1000 µm	3000 ms	1	300 ms	1
2	0 µm	0 ms	0	0 ms	0
3	0 µm	0 ms	0	0 ms	0

Graph: Position (µm) vs. t (ms)

Buttons: Chain, Preview Motion, Save Configuration, IDLE, Enable, SW Trig. Test, HW Trig. Test

Annotations: Slide to increase number of motions, Chain button links motions, Chain on last motion will link it back to first, Updates graph, Save to permanent memory, Activates Kinematic mode so motion can be triggered

Terminal: [CRI] Updating PC Tuning

Modbus communication frequency: active errors

Put motor to sleep, clears errors

Stator 22 °C, Driver 16 °C, Supply Voltage 5.14 V

Status: Connected

Configuring the Kinematic Controller Over Modbus

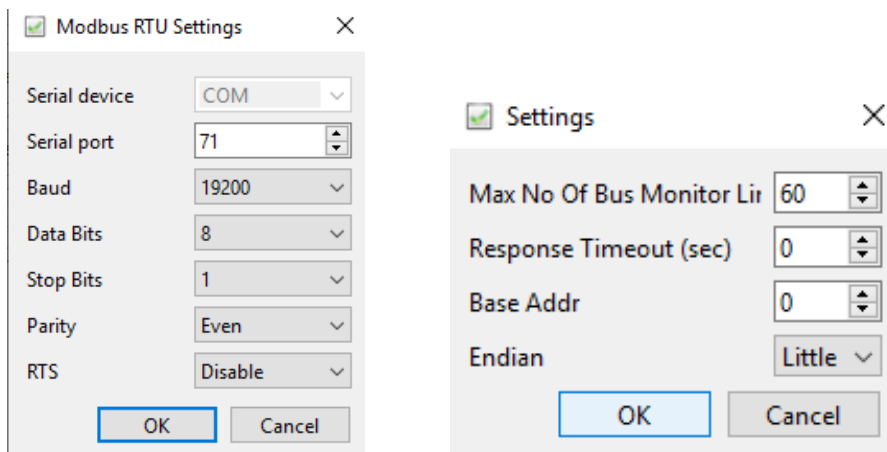
For full details regarding the MODBUS message framing see the Orca Series MODBUS User Guide, with can be found at <https://irisdynamics.com/downloads>

The general format of messages is:

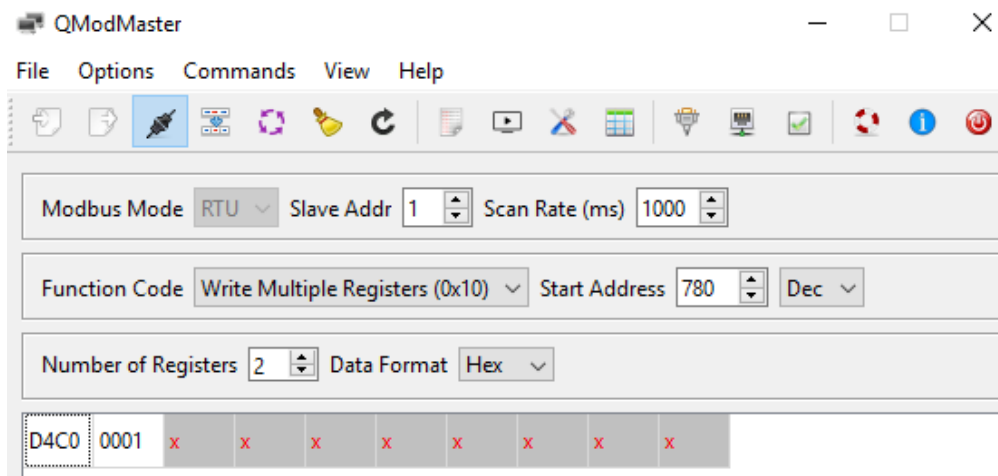
Server Address | Message Type | Start Address | Data | CRC

Example of QModMaster Setup

In the following sections examples of messages being sent using the QModMaster program which automatically calculates the CRC bytes. It can be found for free download here https://download.cnet.com/QModMaster/3000-2085_4-75819467.html. The setting can be configured as follows using your RS422 COM port.

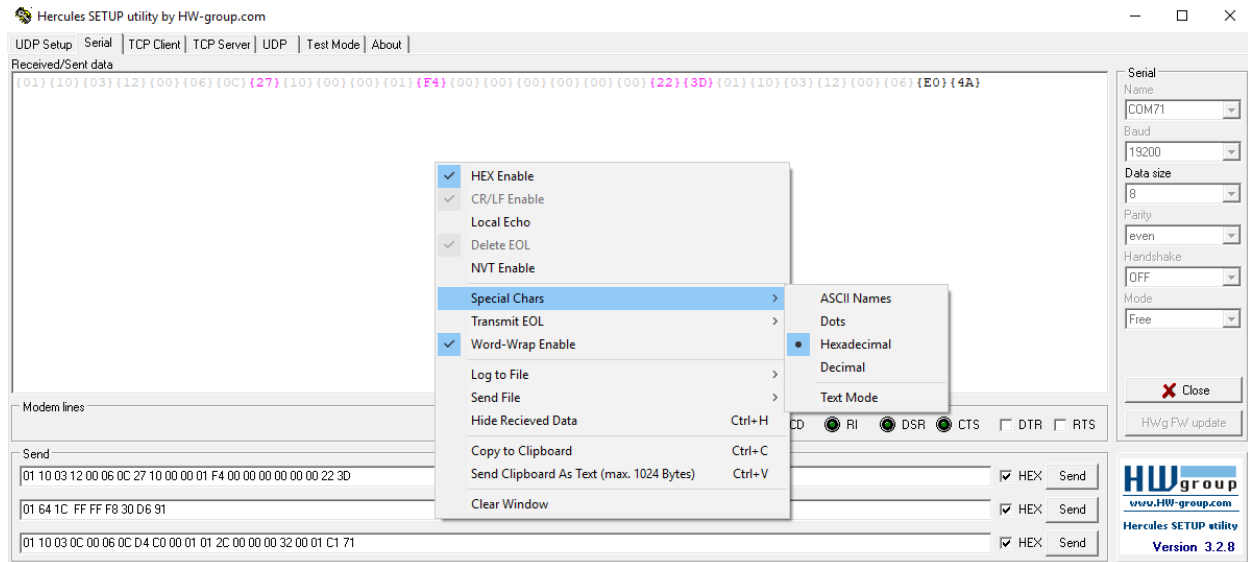


The slave address, function code, start address, and number of registers are all configured from the GUI and the frame at the bottom is only needed



Example Set up Using Hercules

If using a different program such as Hercules the following set up can be used:



CRC Bytes

The CRC value is calculated based on the previous bytes in the message. When the bytes of the message are changed the CRC value will change, every time the same message is sent the CRC value will remain the same.

The CRC bytes can be calculated using online tools such as http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

The polynomial used for generating the 16-bit CRC value for MODBUS communications is specified to be 0xA001.

MODBUS protocol specifies sending the lower CRC byte first followed by the high byte.

Relevant Registers

Registers addresses of interest are shown in the table below.

Register Name	Register Address	Description
CTRL_REG_2	2	For saving configurations
CTRL_REG_3	3	To change the mode
KIN_SW_TRIG	9	Trigger specific motions, or chains of motions
KIN_CONFIG	779	Set the number of motions
KIN_MOTION_0	780	Series of 6 consecutive registers to configure the position, time, type, delay, and chaining of motion 0
KIN_MOTION_1	786	Series of 6 consecutive registers to configure the position, time, type, delay, and chaining of motion 1
KIN_MOTION_n	780 + 6n	Series of 6 consecutive registers to configure the position, time, type, delay, and chaining of motion n

Reading a Register

A good way to test that the MODBUS frames are being formatted correctly and that the correct port is being used is to read a register such as the voltage register (Address 338). This will return the voltage supplied to the motor in mV.

Send Message:

01	03	01	52	00	01	24	27
Server Addr	Read Code	Start Address (338)	Number of Registers	CRC Low	CRC High		

Received Message: (Will differ depending on voltage to motor)

01	03	02	5F	AD	40	09
Server Addr	Read Code	# Data Bytes	Register Value (24493)	CRC Low	CRC High	

Setting the Number of Kinematic Motions

Choose the number of motions you are going to configure. Each motion configured describes a single movement to the specified position. For example, reciprocating back and forth between two positions requires two configured motions. Writing a 1 will indicate two motions.

Use a write single register MODBUS message to write the # of motions – 1 to the KIN_CONFIG register (779).

Set 2 motions: 01 06 03 0B 00 01 39 8C

Set 4 motions: 01 06 03 0B 00 03 B8 4D

Set 16 motions: 01 06 03 0B 00 0F B8 48

Setting the Target Positions

The position is commanded in micrometers (μm) and uses 32 bits. It is split into two 16-bit registers using little endian, so the low byte is written first.

120000 = 0x0001 0xD4C0

KIN_MOTION_0 + 0 = 0xD4C0

KIN_MOTION_0 + 1 = 0x0001

Write multiple registers function code must be used to write to both registers at once. Motions 0 is configured using registers 780 – 785. Motion 1 configuration start at register 786 – 791

Write position 120000 μm to Motion 0: 01 10 03 0C 00 02 04 D4 C0 00 01 1F 06

Write position 120000 μm to Motion 1: 01 10 03 12 00 02 04 D4 C0 00 01 9F 86

Write position 50000 μm to Motion 1: 01 10 03 12 00 02 04 C3 50 00 00 5B DF

Setting the Motion Duration

The time it takes to reach the specified position is specified in milliseconds (ms) and uses 32 bits. It is split into two 16-bit registers with the low byte written first.

Write time 1000 ms to Motion 0: 01 10 03 0E 00 02 04 03 E8 00 00 E6 A3

Write time 10000 ms to Motion 3: 01 10 03 1A 00 02 04 27 10 00 00 6D 5D

If setting a time less than 65535 this can also be accomplished with a write single register frame.

Write time 1000 ms to Motion 0: 01 06 03 14 03 E8 C9 34

Setting the Chain Delay

The next byte is the delay between chained motions which is also in milliseconds (ms) but is a single 16 bit register. If motions are not chained the delay is ignored.

Write delay 100 ms after Motion 0: 01 06 03 10 00 64 89 A0

Write delay 500 ms after Motion 1: 01 06 03 16 01 F4 68 5D

Write delay 2000 ms after Motion 1:

Enabling Chaining and Setting the Motion Type

The type and chain values are both written to the same 16-bit register. The lowest byte enables or disables the chain and the next two bits. If a continuous motion loop is desired, all motions should be chained and the number of motions must match the number of chained motions.

Enable chain, type 0 for Motion 0: 01 06 03 11 00 01 18 4B

Disable chain, type 3 for Motion 0: 01 06 03 11 00 06 59 89

Enable chain, type 2 for Motion 4: 01 06 03 29 00 05 98 45

Configuring an Entire Motion in a Single Frame

It is also possible to set all configurations in a single message.

Motion 0 to move to 120000 μm over 300 ms, delay for 50 ms and chain to motion 1:

01 10 03 0C 00 06 0C D4 C0 00 01 01 2C 00 00 00 32 00 01 71 C1

Repeat this process for each motion you want to configure. For this example, another motion command will be given.

Motion 1 to move to 10000 μm over 500 ms, delay for 0 ms and stop without chaining:

01 10 03 12 00 06 0C 27 10 00 00 01 F4 00 00 00 00 00 00 22 3D

Motion 2 to move to 50000 μm over 150 ms, delay for 1000 ms and chain to motion 3:

01 10 03 18 00 06 0C C3 50 00 00 00 96 00 00 03 E8 00 01 41 D0

Saving Kinematic Configurations

If the configuration is complete, save the kinematic controller settings to permanent memory by writing the *motion_config_save_flag* (0x80) to CTRL_REG_2.

Use the write single register MODBUS command to write the save flag.

Save kinematic configuration: 01 06 00 02 00 80 29 AA

After configuration your motions will now be retained through power cycling. They can be triggered through the KIN_SW_TRIGGER at any time when kinematic mode is enabled, provided the motor has been placed into kinematic mode.

Enabling and Disabling Kinematic Mode

Entering and leaving kinematic mode is done by writing to control register 3. Kinematic mode is mode 5. Sleep mode is mode 1, it will disable kinematic motions and put the motor into a damping state.

Enter Kinematic Mode: 01 06 00 03 00 05 B9 C9

Enter Sleep Mode: 01 06 00 03 00 00 79 CA

Triggering a Kinematic Motion

Trigger your motions using the software trigger register.

To test the configuration, use a write single register MODBUS message to write the motion ID you want to play to the KIN_SW_TRIGGER register. This will also play any chained motions. If all motions are chained, entering kinematic mode will automatically trigger the movement sequence.

Trigger Motion 0: 01 06 00 09 00 00 59 C8

Trigger Motion 1: 01 06 00 09 00 01 98 08

Trigger Motion 8: 01 06 00 09 00 08 58 0E

Clear Errors

Entering sleep mode will clear any motor errors, they can also be cleared by writing 2 to CTRL_REG_0. This will keep the motor in the same mode while any clearable errors are removed.

Clear Errors: 01 06 00 00 00 02 08 0B

Enabling Hardware Triggering

It is recommended that enabling hardware triggering is performed through the Orca Series GUI interface as it disables MODBUS communications.

Troubleshooting

With the motor connected to IrisControls4 (see the Orca Series Motor Reference Manual if you have not done this step yet). Go to the Modbus page.

Messages not being received:

- Bus Message counter is not increasing when a message is sent:
 - Ensure that the com port for the RS422 MODBUS cable is being used for communication and that it is connected both the splitter and the computer.
 - Ensure the MODBUS settings are as follows:
Mode: RTU Baud: 19200 Data Bits: 8 Stop Bits: 1 Parity: Even
- Bus Message counter is increasing but the Server Message counter is not:
 - Hardware triggering mode must be disabled to communicate over MODBUS.
 - Message improperly formatted.

Motor not moving:

- Motor has errors:
 - Any errors can be seen from the Orca's integrated GUI or by reading register ERROR_0 (432). See more details about errors in the Orca Motor Reference Manual
 - Errors can be cleared by sending a [clear errors command](#) or by putting the motor to [sleep mode](#).
- Position controller tuning not set:
 - The position controller must have at least a maximum force (F_{max}) and a P value set in order track the target position. These values must both be large enough for the controller to reach the target. On the Orca Series GUI's Position page, the target position and the current position of the motor are plotted.
- Motor is outputting force but staying at one end of its travel:
 - When the motor is powered on it takes its current position as the zero position. As the shaft is moved toward the cable end it re-zeros itself. The motor should be at the desired zero position when powered on or moved to the desired zero position before entering kinematic mode.
 - This will also be the result if the target position is outside the possible range of movement.

Motor is overshooting position targets:

- Target position on Position page is going past target position values:
 - Motion types 0 and 1 aim to provide minimum power and minimum jerk movements while considering the velocity of the shaft. For this reason, if the shaft is moving quickly it might overshoot a target position to increase the smoothness of the motion. If this is undesirable behaviour, types 2 or 3 can be used which always assume a zero initial velocity when calculating motion targets.
- The target position on the Position page is reaching the correct targets but the motor position is not following:
 - The position controller tuner might need to be adjusted to ensure it is using adequate force to reach the target position. These can be adjusted through the Orca Series GUI or by writing to the corresponding position controller tuning registers (See Orca Series Motor Reference Manual).